Intelligent Urban Traffic

Anastacia Pidgurska, Peter Nikolyuk

***Abstract*—**This investigation aims to create the algorithm that allows to find optimal routes for vehicles in cities where traffic jams are an acute problem. It's necessary to consider that an urgency of the problem is increased day by day. The situation is exacerbated in connection with the constant increasing of the cars number on the city streets. In the meantime, streets and intersections throughput stay virtually unchanged. To solve the formulated problem, logistic software algorithms are used to find the best routes. In this respect the most acceptable for decision of formulated problem is A*-algorithm.

Technically, the flows` control process of vehicles can be done by obtaining information from sensors that are mounted at city intersections. Based on the collected data, a real-time interactive interplay between the Traffic Management Center (TMS) and each vehicle is carried out. The TMS transmits to each driver voice commands regarding the route to the destination, specified by the driver, as in normal GPS navigation. However, the main difference consists in choice of a criterion for optimality – our program chooses as such criterion a time of trip (t-optimality criterion), but usual GPS navigator takes into consideration geometrical factor (g-optimality criterion). The t-optimality criterion is realized by introducing special dynamic scales for the edges of the multigraph that simulates the city's transportation network.

Therefore, the main purpose of this investigation is to lay out t-optimal routes for all vehicles who had ordered such routes. As a result, it will lead to optimization of used transport arteries throughout the city and to a decrease congestion numbers, to synchronization of the vehicles` flows – traffic will move to a new quality level.

***Keywords*—** traffic management center, multigraph, A*-algorithm, t-optimality criterion, g-optimality criterion, GPS navigation.

I. INTRODUCTION

Traffic problem in every metropolis in the world is extremely typical today, but unfortunately is still not resolved. The problem is far from new, but there are no real prospects for solving it. That`s why the author focused on the aim to create the algorithm, which will really give possibility to improve the quality of traffic in cities.

The algorithm is tuned to work in real-time mode, tracking traffic changes and responding appropriately to them when laying t-optimal routes. Laying out such routes works according to the following scheme:

1. Every city intersection will equipped with special sensors – input and output ones. The first sensors are registered cars driving on the road between adjacent intersections; the second type of such devices are registered cars, which leave it. In consequence of this process the entire transport network is under the control of TMS.
2. The interaction between TMS and driver, who uses GPS navigator, is implemented by means of GPS-trackers and as a result, the vehicle driver receives instructions as to choosing of the best route taking into account the latest traffic situation. In other words, car drivers obtain information in regime of real time. On the contrary, in the case with modern GPS navigation drivers obtain the information behind time, when traffic jams on cities streets are already exist.

Anastacia Pidgurska, Vasyl Stus Donetsk National University, Ukraine
Peter Nikolyuk, Supervisor, Vasyl Stus Donetsk National University, Ukraine

II. Analytical Review of the Literature

The study [1] is an overview fundamental work concerning an analysis of modern types of intelligent transportation systems. A wide variety of issues related to the visualization of urban traffic data is addressed. One of the most significant methods of visualizing such data is to use graphs that give possibility to visually represent the topology and geometry of urban areas (in our study the use of graphs and related algorithms there is just the main method of research). The authors also conduct a generalized analysis of a dynamic urban flows mobility, which is as follows:

- Urban traffic flows and their monitoring;
- Human dynamics in an urban environment;
- Road traffic incidents;
- Air pollution.

A striking example of visualization is Fig. 1, where traffic congestion levels is shown – red color represents the highest congestion of the urban network [1].

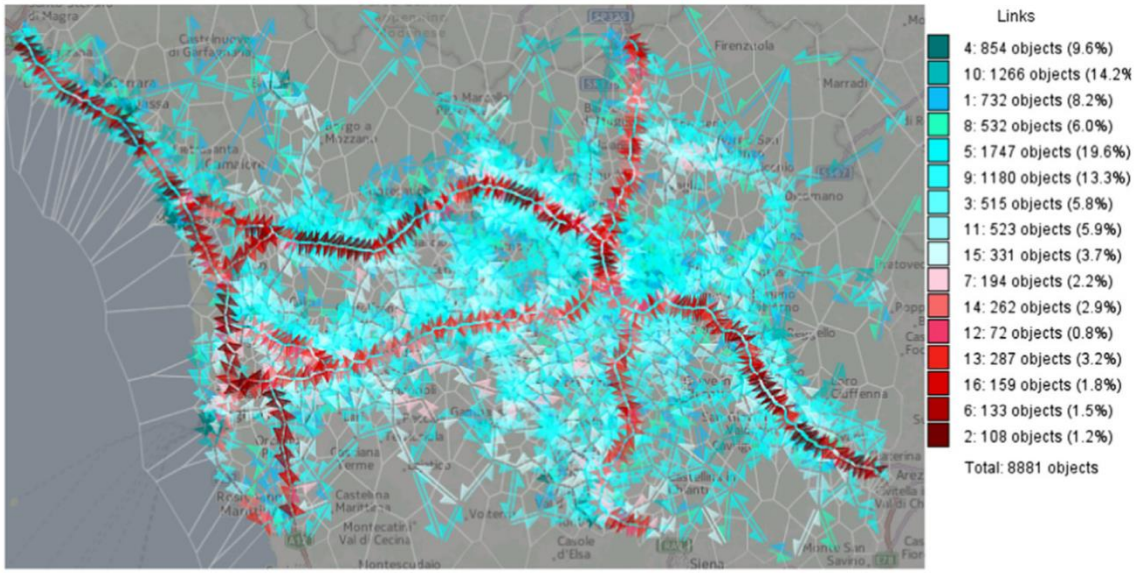


Fig. 1 Visualization of the traffic flows in terms of the vehicle speed value on urban roads (red colors correspond to the transport highways, where the speed is the lowest [1])

A characteristic feature of urban transport network research is using of Vehicular Ad-hoc Networks (VANETs) technology, described in detail in [2]. This technology is used to collect traffic congestion data and transmit it to the TMS. An important applying of VANETs technology is feedback-based navigation. It uses special cars (sensors vehicles) to collect information about the situation on the city's roads and transmits this information to the TMS.

The range of works is devoted to use of the neural networks for an analysis and forecasting of the traffic situations on the transport arteries of cities. In this respect, the authors of work [3] had developed a model that allows to do short-term forecasts (depth of the forecast consists of 5-10 minutes) regarding the behavior of traffic flows on city highways. The study uses the sorting genetic algorithm of type II (NSGA-II), which exploits two important factors – traffic estimation and optimization parameters. A similar previous study had been conducted in [4]. Here Kalman filter is used to predict congestion of urban transport arteries. The technology of "connected cars" is used – such modern vehicles, on which special devices is installed that allow them to interact with urban roadside infrastructure. As a result of such "cooperation" it is possible to estimate various parameters of traffic flows – average speed of traffic, idle time at crossroads, travel time, etc. The algorithm does not use data obtained from other similar

techniques from looped detectors and video cameras, but only data from "connected cars", which proves the cost-effectiveness of this approach.

Effective use of the data collected from various types of road sensors is a significant problem in connection with the loss of some data transmitted to the TMS. This problem is described in an article [5] that proposes a method of data transmission with a high degree of reliability and efficiency. Compared to other technologies of this kind, the quality of data transmission is increased by 87%. The authors introduce a special decomposition matrix that uses the data flow compression principle to calculate traffic metrics.

Traffic forecasting is an important aspect in many areas related to city traffic. The authors of an investigation [6] pay attention to the topic, where three models are selected by analyzing the huge number of methods that make short-term forecasts about the traffic situation. Italian transport networks are investigated using a nonparametric K-neural regression model. Weekly and monthly variations of traffic parameters based on averaged data of this type were studied. Article [7] investigates a problem of integrated traffic control over transport networks by using a macroscopic model, which is a mathematical model of the traffic flow of a vehicle. The density, intensity and speed of the car flow are analyzed. An integrated control algorithm is proposed to solve the problem. The essence of the algorithm is to choose the best routes in the urban transport network.

International inventions, to which the author of this investigation pays great attention, plays a significant role in the problem of solving the fundamental issues of road traffic. In this regard, the prototype of our study is the invention [8], which is a technology related to a use of the mobile phones by drivers of vehicles. At the intersections of a city special devices are installed to record traffic flows and transmit the collected information to the TMS. The information then will sent to each driver via the Internet in order to choose the best route for them. The invention is solved the problem of traffic optimization by estimating a flow of vehicles through each city intersection [9]. The system operates in real time mode and therefore dynamically controls and regulates traffic on urban routes by creating a green wave mode. In [10] the method of time-series analysis of Markov hidden chain is applied. As the main criteria are used optimizing travel time and vehicle speed, that as a result leads to finding the best option trip.

Interaction between cars on urban routes (and not only) plays an important role in organizing of a coordinated behavior of a large number of dynamic objects, especially when they are crowded, as is a case in metropolis cities. The problem of such interaction based on the standard IEEE 802.11p is considered in [11]. «Car-to-Car» interaction should be provided by a reliable communication system, which is the standard mentioned above.

Transport networks work in different modes [12]. Knowledge of these modes and transitions between them is an important factor that determines an essence of the transport network operation. Overload of city road network leads to congestion. In order to avoid such problems, it is necessary to study in detail the principles of the transport network operation. This work examines the statistical and dynamic transitions between different modes of traffic. Investigation [13] is addressed to the specific problem of improving traffic in an urban environment, where congestion is often happened due to bad traffic management. A similar problem is addressed in the study [14], which is an element of technology suitable for practical use. An emphasis has been done on four areas – graph theory, intellectual A*-algorithm, information technology and technical means of the vehicles registration (sensors). A combination of these components makes a real step in a solving of a major problem of each big city – the problem of traffic jams. A real technology for solving the mentioned problem provide authors of an invention [15] too. This innovation relates to an intelligent control system of traffic lights by bilateral exchange of information with TMS.

III. Object, Subject And Methods Of Research

This investigation is focused on solving of significant problem of big cities – the problem of traffic jams. Therefore, the object of this study is the transport networks of the megapolis cities. The problem to be solved – the subject of this study – is congestion on urban transport networks. Creating an algorithm for laying t-optimal routes for each vehicle involved in traffic is the main goal of the work. Research methods: 1) methods of theory for modeling of the complex systems; 2) methods of a graph theory; 3). Java technologies; 4). optimization methods and A*-algorithm.

IV. Work Results

Use either SI (MKS) or CGS as primary units. (SI units are strongly encouraged.) English un Let`s consider part of Toledo (Canada) transport network, which is presented in Fig.2. It is necessary to create a transport network model for this part of the city, taking into account number of lines on each section of the road. All this have been done in Fig. 3.

Fig. 2 A microdistrict of Toledo (Canada). The dotted line highlights a part of the area modeled by the graph shown in the following figure

As can be seen (Fig. 3), the presented graph simulates not only the road between adjacent intersections, but also individual lines, that is fundamentally significant point, since a congestion of the different lines is usually essentially different. Therefore, it is need register them separately. The multigraph presented in Fig. 3 is weighted – each edge receives a changing weight (this will be discussed further). The nodes of the graph simulate the roads intersections, and the directed arcs (edges) symbolize lines.

Labels “START” and “FINISH“ symbolize the starting and ending positions of the car that had ordered a route between the indicated positions. In the center of Fig. 3 there is designations of type b7_6, a7_6, a6_7, b6_7 and c6_7. How such designations should be understood? First, the traffic lines are denoted by a, b, c, d,…, starting from a center of the road, as shown in Fig. 4.

So, in particular, the entry b7_6 denotes the middle line extending from node (intersection) 7 to node 6. Record c6_7 represents the extreme right line (with only three lines in one direction of the road) extending from intersection 6 to the intersection 7. Similarly, other inscriptions of this type are deciphered. To plot t-optimal route in graph presented in Fig.3, it is necessary to register the vehicles located on each of the lines. The process of such registration can be understood by looking at Fig. 4.

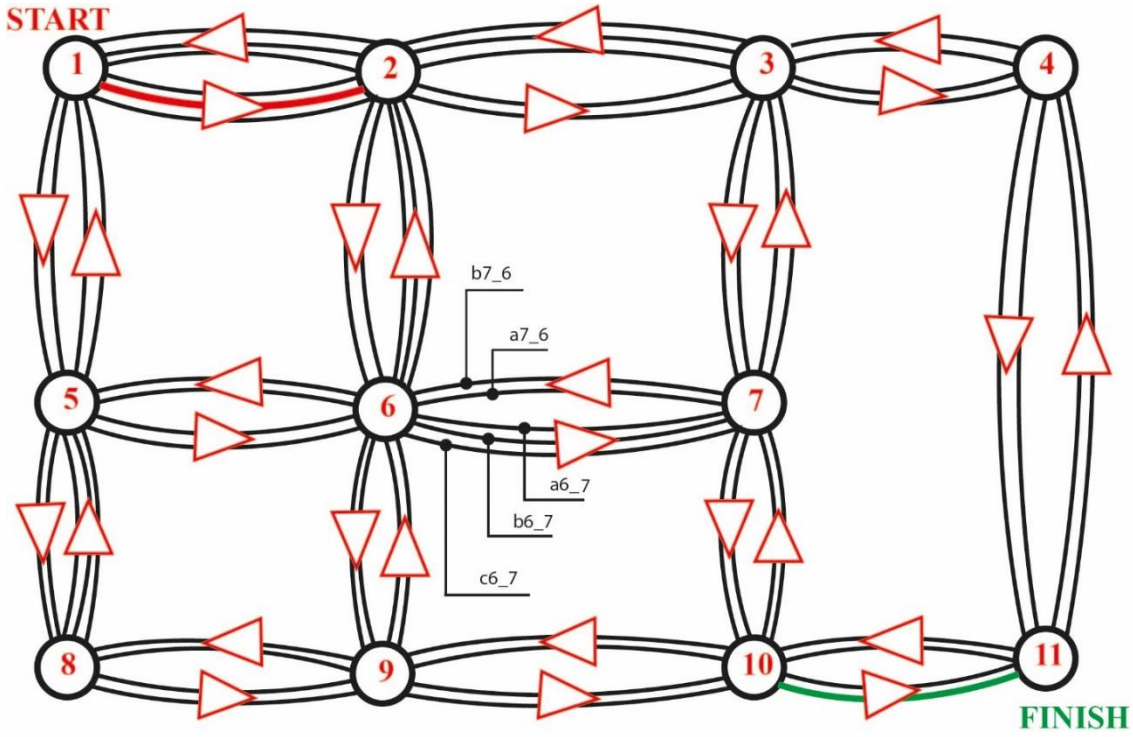


Fig. 3 A weighted planar oriented multigraph that simulates the city's transportation network

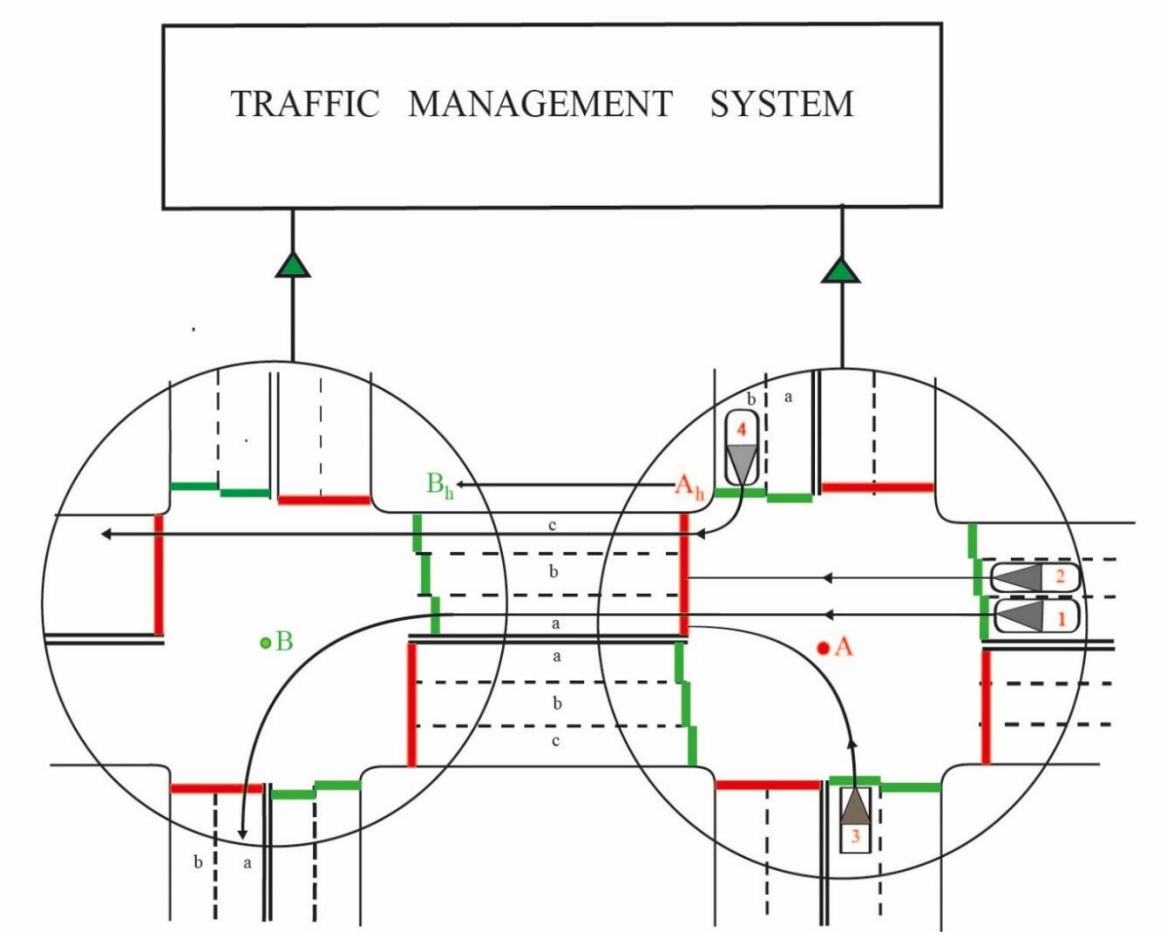
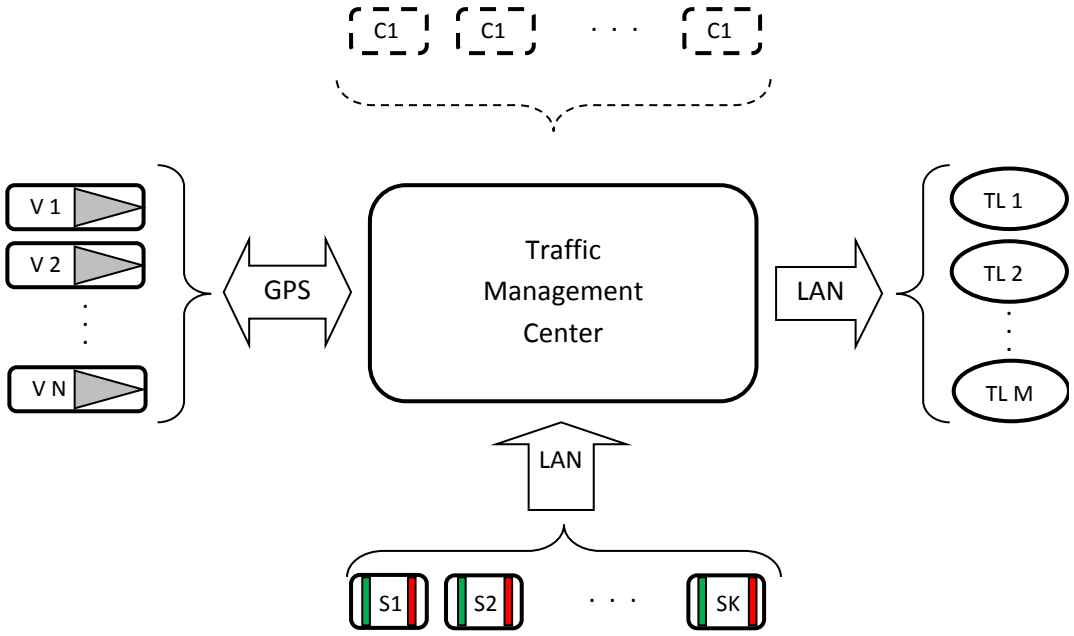


Fig. 4 . Two adjacent crossroads – A and B. The vehicles driving to the section of the road between these intersections are shown. The red bars are the input sensors. The output sensors are represented as three green stripes in a row – each strip is equipped with its own sensor. It is shown the routes of the vehicles at number 1 and 4. The data from input and output sensors is coming to TMS.

The key item on Fig.4 is the presence of two types of sensors – input and output ones. These are so-called piezoelectric sensors, such as the Roadtrax BL type [13]), which respond on pressure caused by wheeled couple of vehicle. These sensors are mounted in a roadway perpendicular to the longitudinal axis of the road immediately behind the stop line. The received impulses are received for registration at the measuring and computing complex (MCC) and then to TMS. In this way, lines between adjacent intersections are under control. Input sensors register all vehicles as a whole, driving to the lane from all possible directions of the intersection A, that is, the total number of the vehicles. These sensors also play a fundamental role in the transmission of updated routes to cars – as soon as the vehicle has crossed the input sensor, the program at the TMS calculates and transmits a new vehicle route. The output sensors register the cars separately, that is, for each individual line of the road. Following, the number of output sensors is equal to the number of lines of the carriageway in one direction. The presence of input and output sensors, makes possible fully register the transport flows. In order to regulate such flows, the t-optimal routes are laid for all vehicles involved in traffic. In large cities, the number of cars on the streets at peak hours can reach up to a million. This means that their flows must be reconciled, synchronized, by the way of finding the optimal route for each vehicle.



V – vehicle,

TL – traffic lights,

S – road sensors,

LAN – local area network (f.e. LoRaWAN),

C – camera.

Fig. 5 . An interaction scheme of the TMC with cars, traffic lights at intersections and sensors. Video cameras are also shown as an additional source of cross-traffic information.

In Fig. 5 schematically presented the processes of interaction between TMS and traffic components – cars, road sensors, traffic lights. Video cameras are used as an additional source of information on urban network congestion.

Technically, the process of optimizing traffic flows consists in transferring data from the MCC to the TMC, where, based on a special program, t-optimal routes of traffic for each vehicle are calculated. Then, the traffic routes are transmitted to each driver on the GPS navigator. As the data is constantly updated, the program, which will be presented below, is constantly searching for t-optimal routes for all vehicles in a metropolis. Here, optimality is determined by fulfilling the following condition:

$$\sum_{h=1}^{f} (N^{i}_{A_hB_h} / (n^{i}_{A_hB_h}))l_{A_hB_h} \to min. \quad (1)$$

Here $N^{i}_{A_hB_h}$ – the number of cars driving on a line i of the road section $A_h \to B_h$; $n^{i}_{A_hB_h}$ – the number of cars leaving the line i; h – an index numbering the sections of the road that make up the whole route. Number f – is the number of sections of road that form the route; $l_{A_hB_h}$ – length of the road section h.

In the Listing 1 the instruction

```
double ra2_1 = (1 + (double) k.nextInt(5)) * L2_1,
```

corresponds presented up condition (1), where ra2_1 is an attitude with the number of vehicles driving on a road between adjacent intersections and those who leaving it.

In expression (1) time t is clearly not included! The question then is: why does this condition correspond to the optimality condition over time? In this connection it is need to consider the relation appearing in the multiplication of the expression (1):

$$N^{i}_{A_hB_h} / n^{i}_{A_hB_h} . \quad (2)$$

The magnitude (2) has a great influence on a duration of a trip. Suppose that the relation (2) is equal three in the middle. This actually means that at each intersection the car is delayed for three periods (cycles) of traffic light switching. If one cycle of the traffic light switching is approximately 70 s, then the delay at each intersection will be 210 s, and we will get a very significant value along the whole route. Ideally, the value (2) should be within a unit: in such case trip time will be reduced by three times! Our algorithm, using (1), searches such route for which value (2) is approximately equal unit. That is why condition (1) is a criterion for optimality over time.

In general, the value of the relation (2) for each section of the road can take a wide range of values. Consider two extreme options:

$$N^{i}_{A_hB_h}/(n^{i}_{A_hB_h}) \; 1, \tag{3}$$

$$N^{i}_{A_hB_h}/(n^{i}_{A_hB_h}) \gg 1. \tag{4}$$

In the case of condition (3), the edge of a graph (in other words, the traffic line between adjacent intersections) is closed to traffic, since cars drive on the line of the road, but they are delayed for a long time on this line (the wait time exceeds several switching cycles traffic light). In fact, this means that such line of the road is closed for traffic, and therefore the program will bypass it laying the optimal route and redirects cars to less load directions. As already mentioned, the other point in case (4) – the corresponding lines of the road are free for traffic, the dynamics of car on them is maximum (or close to that), and therefore the corresponding links of the road between the intersections will be included in the calculated t-optimal route.

The computer program below (Listing 1) implements an A*-algorithm which is inherently optimization one. The program contains three classes – Astar, Node and Edge. In the field of the first of the named classes all constructors of the nodes of the graph presented in Fig. 3 are implemented. The initGraph () method of the same class contains the Random () constructor, which is used to generate car line downloads. The parameters of this constructor are chosen in such way that the traffic congestion (values of type ra2_1) lie within 1 ÷ 10, which corresponds to the real situation on city streets. Type constructors

```
b1_2.adjacencies = new Edge[]{
new Edge(a2_3, ra2_3),
new Edge(b2_3, rb2_3),  };
```
(5)

describes the interaction between the edges of a graph (in other words, between lines). It is noteworthy that it adheres to the traffic rules (TR). Indeed, if we analyze (5), it can be seen that from the line b1_2, that is, the middle line, it is possible to cross the intersection 2 only directly (see Fig. 3), to get into the lines a2_3 and b2_3, but not to turn left or right. And in such manner throughout the graph for all lines. Thus, the program completely reproduces the real situation on the city roads and can serve as a guide for drivers. It is need to note that the initGraph () method of the Astar class includes all constructors of type (5). However, the number of such constructors is equal to the number of edges in the graph (last number is equal 65). Accordingly, the length of this method is quite large and therefore we cannot give it completely. We give only constructors for two nodes. It is important to note that A* algorithm operates on quantities $f(n)=g(n)+h(n)$, where $g(n)$ is a distance in the graph from the starting vertex to the current one. The $h(n)$ is so called heuristic distance, measured in a straight line from the

current vertex n to the end (in the Astar class field, the node constructors include as arguments the heuristic spectrum measured with respect to the specified starting and finishing positions shown in Fig. 3).

Listing 1.

```
import java.util.*;
public class Astar {
        static Node a1_2 = new Node("a1_2-Liberty avenue", 962);
        static Node b1_2 = new Node("b1_2-Liberty avenue", 962);
        static Node c1_2 = new Node("c1_2-Liberty avenue", 962);
        static Node a1_5 = new Node("a1_5-Freedom street", 1002);
        static Node b1_5 = new Node("b1_5-Freedom street", 1002);
        static Node a2_1 = new Node("a2_1-Liberty avenue", 962);
        static Node b2_1 = new Node("b2_1-Liberty avenue", 962);
        static Node c2_1 = new Node("c2_1-Liberty avenue", 962);
        static Node a2_6 = new Node("a2_6-Heroes street", 733);
        static Node b2_6 = new Node("b2_6-Heroes street", 733);
        static Node a2_3 = new Node("a2_3-Broad street", 753);
        static Node b2_3 = new Node("b2_3-Broad street", 753);
        static Node a3_2 = new Node("a3_2-Broad street", 753);
        static Node b3_2 = new Node("b3_2-Broad street", 753);
        static Node c3_2 = new Node("c3_2-Broad street", 753);
        static Node a3_4 = new Node("a3_4-Short street", 658);
        static Node b3_4 = new Node("b3_4-Short street", 658);
        static Node a3_7 = new Node("a3_7-Valley street", 504);
        static Node b3_7 = new Node("b3_7-Valley street", 504);
        static Node a4_3 = new Node("a4_3-Short street", 658);
        static Node b4_3 = new Node("b4_3-Short street", 658);
        static Node a4_11 = new Node("a4_11-Long street", 325);
        static Node b4_11 = new Node("b4_11-Long street", 325);
        static Node a5_1 = new Node("a5_1-Freedom street", 1002);
        static Node b5_1 = new Node("b5_1-Freedom street", 1002);
        static Node a5_6 = new Node("a5_6-Middle street", 777);
        static Node b5_6 = new Node("b5_6-Middle street", 777);
        static Node a5_8 = new Node("a5_8-Victory street", 887);
        static Node b5_8 = new Node("b5_8-Victory street", 887);
        static Node a6_5 = new Node("a6_5-Middle street", 887);
        static Node b6_5 = new Node("b6_5-Middle street", 887);
        static Node a6_2 = new Node("a6_2-Heroes street", 733);
        static Node b6_2 = new Node("b6_2-Heroes street", 733);
        static Node c6_2 = new Node("c6_2-Heroes street", 733);
        static Node a6_7 = new Node("a6_7-Happy street", 523);
        static Node b6_7 = new Node("b6_7-Happy street", 523);
        static Node c6_7 = new Node("c6_7-Happy street", 523);
        static Node a6_9 = new Node("a6_9-Joy street", 569);
        static Node b6_9 = new Node("b6_9-Joy street", 569);
        static Node a7_6 = new Node("a7_6-Happy street", 523);
        static Node b7_6 = new Node("b7_6-Happy street", 523);
        static Node a7_3 = new Node("a7_3-Valley street", 504);
        static Node b7_3 = new Node("b7_3-Valley street", 504);
        static Node a7_10 = new Node("a7_10-Trump street", 289);
        static Node b7_10 = new Node("b7_10-Trump street", 289);
        static Node a8_5 = new Node("a8_5-Victory street", 887);
        static Node b8_5 = new Node("b8_5-Victory street", 887);
        static Node c8_5 = new Node("c8_5-Victory street", 887);
        static Node a8_9 = new Node("a8_9-Lincoln street", 713);
        static Node b8_9 = new Node("b8_9-Lincoln street", 713);
        static Node a9_6 = new Node("a9_6-Joy street", 569);
        static Node b9_6 = new Node("b9_6-Joy street", 569);
        static Node b9_8 = new Node("b9_8-Lincoln street", 713);
        static Node a9_8 = new Node("a9_8-Lincoln street", 713);
        static Node a9_10 = new Node("a9_10-Kennedy street", 400);
        static Node b9_10 = new Node("b9_10-Kennedy street", 400);
        static Node a10_7 = new Node("a10_7-Trump street", 289);
        static Node b10_7 = new Node("b10_7-Trump street", 289);
        static Node a10_9 = new Node("a10_9-Kennedy street", 400);
```

```
            static Node b10_9 = new Node("b10_9-Kennedy street", 400);
            static Node a10_11 = new Node("a10_11-Apple street", 125);
            static Node b10_11 = new Node("b10_11-Apple street", 125);
            static Node a11_10 = new Node("a11_10-Apple street", 125);
            static Node b11_10 = new Node("b11_10-Apple street", 125);
            static Node a11_4 = new Node("a11_4-Long street", 325);
            static Node b11_4 = new Node("b11_4-Long street", 325);
            static Node nodes[] = new Node[]{
    a1_2, b1_2,c1_2, a1_5, b1_5, a2_1, b2_1,c2_1,a2_6,b2_6, a2_3, b2_3,
    a3_2, b3_2,c3_2,a3_4, b3_4,a3_7,b3_7,a4_3,b4_3,a4_11,b4_11,a5_1,
    b5_1,a5_6,b5_6,a5_8,b5_8,a6_5,b6_5,a6_2,b6_2,c6_2,a6_7,b6_7,c6_7,
    a6_9,b6_9,a7_6,b7_6,a7_3,b7_3,a7_10,b7_10,a8_5,b8_5,c8_5,a8_9,
    b8_9,a9_6,b9_6,b9_8,a9_8,a9_10,b9_10,a10_7,b10_7,a10_9,b10_9,a10_11,
                    b10_11, a11_10,b11_10,a11_4,b11_4  };
            public static void initGraph() {
                Random k = new Random();
                int L2_1 = 320;
                double ra2_1 = (1 + (double) k.nextInt(5)) * L2_1;
                double rb2_1 = (1 + (double) k.nextInt(5))  * L2_1;
                double rc2_1 = (1 + (double) k.nextInt(5)) * L2_1;
                a1_2.adjacencies = new Edge[]{
                        new Edge(a2_1, ra2_1),
                        new Edge(b2_1, rb2_1),
                        new Edge(c2_1, rc2_1),};
                          int L2_3 = 340;
                double ra2_3 = (1 + (double) k.nextInt(5)) * L2_3;
                double rb2_3 = ((1 + (double) k.nextInt(5))  * L2_3;
                b1_2.adjacencies = new Edge[]{
                        new Edge(a2_3, ra2_3),
                        new Edge(b2_3, rb2_3),
```

. . . . . .

After that, in the main method of the program and in the classes Node and Edge the route between the lines b1_2 and b10_11 (in Fig. 3 these lines are highlighted) is implemented.

```
              public static void main(String[] args) {
            initGraph();
            AstarSearch(b5_6, b10_11);
            List<Node> path = printPath(b10_11);
            System.out.println("Path: " + path);
            while (path.size() > 1) {
                Node node = path.get(1);
                node.parent = null;
                for (Node n : nodes) {
                    n.parent = null;
                initGraph();
                AstarSearch(path.get(1), b10_11);
                path = printPath(b10_11);
                System.out.println("Path: " + path);
            }
                    }
        public static List<Node> printPath(Node target) {
            ArrayList<Node> path = new ArrayList<Node>();
            for (Node node = target; node != null; node = node.parent) {
                path.add(node);
            }
            Collections.reverse(path);
            return path;
        }
        public static void AstarSearch(Node source, Node goal) {
            Set<Node> explored = new HashSet<Node>();
            PriorityQueue<Node> queue = new PriorityQueue<Node>(100, new
Comparator<Node>() {
```

```
                    //override compare method
                    public int compare(Node i, Node j) {
                        if (i.f_scores > j.f_scores) {
                            return 1;
                        } else if (i.f_scores < j.f_scores) {
                            return -1;
                        } else {
                            return 0;
                        }
                    }
                });
                source.g_scores = 0;
                queue.add(source);
                boolean found = false;
                while ((!queue.isEmpty()) && (!found)) {
                    Node current = queue.poll();
                    explored.add(current);
                    if (current.value.equals(goal.value)) {
                        found = true;
                    }
                        for (Edge e : current.adjacencies) {
                        Node child = e.target;
                        double cost = e.cost;
                        double temp_g_scores = current.g_scores + cost;
                        double temp_f_scores = temp_g_scores +
child.h_scores;
                          if ((explored.contains(child)) &&
                                (temp_f_scores >= child.f_scores)) {
                            continue;
                        } else if ((!queue.contains(child)) ||
                                (temp_f_scores < child.f_scores)) {
                            child.parent = current;
                            child.g_scores = temp_g_scores;
                            child.f_scores = temp_f_scores;
                            if (queue.contains(child)) {
                                queue.remove(child);
                            }
                            queue.add(child);
                        }
                    }
                }
            }
        }
        class Node {
            public final String value;
            public double g_scores;
            public final double h_scores;
            public double f_scores = 0;
            public Edge[] adjacencies;
            public Node parent;
            public Node(String val, double hVal) {
                value = val;
                h_scores = hVal;
            }
            public String toString() {
                return value;
            }
        }
        class Edge {
            public final double cost;
            public final Node target;
            public Edge(Node targetNode, double costVal) {
                target = targetNode;
                cost = costVal;
            }
```

```
        public String toString() {
            return "Edge(" + target + ", " + cost + ")";
        }
    }
```

Running the program, we can get the result below (another program launch usually gives other data due to the action of the random numbers generator):

Path1: [b1_2-Liberty avenue, **b2_3-Broad street**, b3_7-Valley street, a7_6-Happy street, a6_9-Joy street, b9_10-Kennedy street, b10_11-Apple street]

Path2: [b2_3-Broad street, **b3_7-Valley street**, a7_6-Happy street, a6_9-Joy street, b9_10-Kennedy street, b10_11-Apple street]

Path3: [b3_7-Valley street, **a7_6-Happy street**, a6_9-Joy street, b9_10-Kennedy street, b10_11-Apple street]

Path4: [a7_6-Happy street, **a6_9-Joy street**, b9_10-Kennedy street, b10_11-Apple street]

Path5: [a6_9-Joy street, **b9_10-Kennedy street**, b10_11-Apple street]

Path6: [b9_10-Kennedy street, **b10_11-Apple street**]

Path7: [b10_11-Apple street]

The result of the program has a row of specific features. The calculated route is a set of steps of type Path1 –Path7. Particularly note that the transitions from one step to another are interconnected and are carried out using the calculation result in the previous step – the program intercepts the second position of the previous calculated route (these key lines are highlighted) and starting from this position calculates a new route. It is implied that the route calculated in the first step will be preserved only *during the journey by the driver of only one line* – due to the high dynamism of urban traffic. As soon as the vehicle has crossed the intersection, the program calculates a new route, starting each time from the second pre-calculated position. For example, Path2 uses the *b2_3-Broad street* as a starting position (which stands second in Path1) to calculate a new route. The program then uses *b3_7-Valley street* as the starting position for calculation of Path3. The following routes are similarly calculated until the final position is reached (in this case, the b10_11-Apple street line). As emphasized above, the transmission of the updated route is initiated by the input sensors – as soon as the car crosses the input sensor, the program on the TMS calculates and transmits a new path to the GPS navigator. Thus, thanks to constantly updated routes, the traffic software has a high degree of dynamism and at the same time is very comfortable for every driver *because the driver is able to adjust to the desired line, as every new instruction comes to him immediately after crossing the intersection.*

As all cars (or the vast majority of them) will move along t-optimal routes, complete synchronization of traffic flows will occur, resulting in fundamentally new quality and, as a consequence, to the disappearance of traffic jams (or their abrupt reduction) in the city transport networks. This circumstance will allow each driver to arrive at his destination in a shortest possible time. Thus, urban traffic has transformed into a fundamentally new state.

Testing of Listing 1program had been performed on graphs with vertices up to 165 and edges up to 445. It is need to note that the complexity of the A*-algorithm is linear O (n), which makes it quite efficient for the tasks with a large amount of input data.

The testing of the described algorithm was carried out by modeling the transport network with the help of the highly efficient AnyLogic 8.5.1 simulation program [16]. The simulation results show a significant increase in the capacity of urban transport networks compared to the standard mode when using conventional GPS navigation.

V. Conclusion

The work has practical nature and aims at a real solution of traffic problem in big cities. According to the results of the investigation, an article [14] had been published.

Model experiments show the full acceptability of the algorithm for the purposes of technical implementation. This circumstance will help get rid of the problems of every major city – congestions – and it will make easier for drivers to follow along stated route. The traffic climate on cities` streets will improve significantly – congestions will not be a great problem!

References

[1] Thiago Sobral, Teresa Galvao and Jose Bornes, Visualization of Urban Mobility Data from Intelligent Transportation Systems, Sensors, V.19, Is.2, P.332 – 360, 2019.

[2] Ahmed Elbery and Hesham Rakha, City-Wide Eco-Routing Navigation Considering Vehicular Communication Impacts, Sensors, V.19, Is.2, P.290 – 311, 2019.

[3] Shiva Rahimipour, Rayehe Moeinfar, S. Mehdi Hashemi, Traffic prediction using a self-adjusted evolutionary neural network, J. Mod. Transport, V. 27, Is.4, P.306–316. 2019.

[4] Azadeh Emami, Majid Sarvi, Saeed Asadi Bagloee, Using Kalman filter algorithm for short-term traffic flow prediction in a connected vehicle environment, J. Mod. Transport, V. 27, Is.3, P.222–232, 2019.

[5] Xianglong Luo, Xue Meng, Wenjuan Gan, and Yonghong Chen, Traffic Data Imputation Algorithm Based on Improved Low-Rank Matrix Decomposition, Journal of Sensors, Article ID 7092713, 11 pages, 2019.

[6] Andrea Pompigna, Federico Rupia, Comparing practice-ready forecast models for weekly and monthly fluctuations of average daily traffic and enhancing accuracy by weighting methods, Journal of Traffic and Transportation Engineering (English Edition), V.5, Is. 4, P. 239-253, 2018.

[7] Hirsh Majid, Chao Lu, Hardy Karim. An integrated approach for dynamic traffic routing and ramp metering using sliding mode control, Journal of Traffic and Transportation Engineering (English Edition), V.5, Is. 2, P. 116-128, 2018.

[8] Pat. CN104064049B China, A kind of intelligent transportation road capacity note broadcasting system, Xu Chunmao Xu Jin; publ.08.03.2017.

[9] Pat. CN110136454 (A) China, Urban artery traffic dynamic green wave signal control system and method based on real-time traffic flow data/ Shu Aibing, Xu Xindong, Xu Leng, Zhang Bin, Liu Chengsheng, Cai Yubao; applicant Traffic Management Res Institute of the Ministry of Public Security; publ.16.08.2019 .

[10] Pat. CN109920250 (A) China, Dynamic prediction intelligent traffic management method for urban road / Zhang Hongtao, Li Tianxue, Hu Haitao, Li Lei, Hao Yan, Chen Qingshan, Zhang Weiling, Chu Peng, Jiang Chunhui, Zhou Wei; applicant Li Tianxue; publ. 21.06.2019.

[11] Tong Wang , Azhar Hussain, Yue Cao, Sangirov Gulomjon . An Improved Channel Estimation Technique for IEEE 802.11p Standard in Vehicular Communications, Sensors, V.19, Is.1, 22p., 2019.

[12] Emmanuel Kidando, Ren Moses, Thobias Sando, Eren Erman Ozguven. An application of Bayesian multilevel model to evaluate variations in stochastic and dynamic transition of traffic conditions, J. Mod. Transport, V. 27, Is.4, P.235–249, 2019.

[13] D.G. Boguto, V.F. Komarov, P.K. Nikolyuk, P.P. Nikolyuk, Intelligent algorithm for management of urban traffic of vehicles, Bulletin of V.Karazin Kharkiv National University, series "Mathematical modeling. Information Technology. Automated Control Systems. ", Is. 38, pp. 4-13, 2018.

[14] D.G.Boguto, K.K. Kadomskiy, P.K. Nikolyuk, A.I. Pidgurska, Algorithm of Intelligent Urban Traffic , Bulletin of V.Karazin Kharkiv National University, series "Mathematical Modeling. Information Technology. Automated Control System. ", Is.42, P.12-25, 2019.

[15] Pat. CN104575066 China, Intelligent terminal based intelligent traffic light system and method/ Xu Chunmao Xu Jin; applicant Shanghai Bolter Digital Technology Co., LTD.; publ.29.04.2015.

[16] Alpha version of Any-Logic 8.5.1 [Electronic resource]. Access mode: https://www.anylogic.com/downloads.